Process of Digital Signature and Cryptography Communication in Network simulator-2 Environment

Dr.Ajit Kumar Singh [1], Mithilesh Kumar Singh Yadav [2]
Professor [1], Assistant Professor [2] Department of CSE
Sherwood College of Engg. Research & Technology, Lucknow
(ajit2504singh [1], mithilesh.mnnit [2])@gmail.com

Abstract— NS2 is an event-driven, object-oriented simulation tool to simulate and analyze dynamic nature of communication networks; it is also a powerful tool to develop new protocols and functions. NS-2 is an open source and very popular network simulation tool. It provides support for TCP/IP protocols suite and many other routing protocols for wired and wireless networks. In NS2, many protocols have been implemented so far at various layer of TCP/IP protocol suite to provide different functions but none provides security functions. Although, some applications require security (Authentication and integrity of message) implementation in NS-2. However, NS-2 does not provide these features till now. In this paper we solve this issue by adding new security module or protocol in NS2. Security module helps us to incorporate functions for integrity of the message and authentication of the sender. We analyze the features of security module in details; discuss the algorithms used, simulation process and implementation of security module on the basis of NS2. Also, the simulation details of sender authentication protocol and the function for the integrity of the message in wired network are introduced. NAM is used to display the process of simulation. The purpose of the module is to introduce Digital Signature features with integrity of message into network simulation program.

Keywords: NS2, Integrity of Message, Sender Authentication, Public Key Cryptography, Encryption/Decryption, RSA Algorithm.

I. INTRODUCTION

NS-2 [3, 13] is an open source, event driven system that is developed using C++ and TCL language [18]. In NS2, researchers can easily add new components and functions to the system to serve their own purposes. The latest version of NS-2 supports most of the standard protocols. You can find protocol from data link layer such as CSMA/CD up to application protocols as FTP, TELNET DNS and HTTP. All these protocols were developed by researchers and later incorporated into standard version of NS-2. In order to analyze security features for the network, we need to add security module into NS-2. So, this paper only describe a way to build and add security module by which we can incorporate key sharing features as well as encryption/decryption features into NS-2. Our approach is to add a new protocol at the network layer, and it is specific to a wired network. We also define new packet format for new security protocols. The new protocols are represented by a class derived from built-in class in NS-2. Within new derived class we add encryption/decryption function for data field in the data packet for sender authentication. We also implement hash function to ensure the integrity of data during transmission. We consider our data as plain text. The encryption/decryption and hash algorithm are RSA [17] and MD5 [17] respectively. The programming languages are C++, TCL, AWK. Environment development requirements are:

II. RELATED WORK

Ns is fairly large. The allinone package requires about 320MB of disk space to build. Building ns from pieces can save some disk space. (If multiple people want to share files in the ns build tree to save space, you may download a , then follow the instruction in its README. There is detailed instruction from CS599b class of USC. You may also find discussions in the ns-users mailing list archive useful.)In NS2, many protocols have been implemented at various layer of TCP/IP protocol suite. For wired network, application layer protocols like telnet, ftp, cbr, http, etc. and transport layer protocol like TCP, UDP, variants of TCP, etc. and network layer protocols like IP, Ping, etc. and Data link layer protocol like CSMA/CD, proposed by the researchers and later these protocols were incorporated in NS2. Similarly, for wireless network many protocols at different layer of TCP/IP protocol suite were proposed like AODV, TORA, DSR, DSDV [6], DYMO [7], etc. Later these protocols became a part of NS2. Recently, researchers are focusing on to propose new protocols for wired and wireless networks [8] and some are trying to enhance or optimize the previously available protocols. But, very less work has been done in the field of incorporating security of data in NS2. Author JIANG Hong with YU Qing-song and LU Hui of East China Normal University, Shanghai, China have proposed their work [5] in this direction. Their schemes enhance the security of Ethernet by a group based MAC key selection protocol (GKSP) for large Ethernet networks are provided. They provided security at the data link layer i.e. security has hop to hop jurisdiction. **What hardware is needed?** To build ns you need a computer and a C++ compiler. We develop ns on several kinds of Unix (FreeBSD, Linux, SunOS, Solaris), so it installs smoothest there, but it should run on an Posix-like computer, possibly with some tweaking. Ns also builds and runs under Windows, see the dedicated . Simple scenarios should run on any reasonable machine, but very large scenarios benefit from large amounts .

In our security module we provide security at the network layer because we can easily implement security for different types of applications and security has end to end jurisdiction. The security module that we developed intend to solve the problem for both sender authentication and integrity of data at network layer, hence incorporating and enhancing the security of data in NS2. We decided to use free available resources, i.e. the same programming platform as NS2 and based on Red Hat Linux.

III. SIMULATION OF WIRED NETWORK

. Message authentication is a service used to verify the integrity of a message. Message authentication assures that the data received are exactly same as sent. Hash function is used to provide message authentication. The hash code value is also called as message Digest. Hash function value is used for message authentication in terms of this: “The sender calculates a hash code value as a function of the bits in the message and transfers both the hash value

1) When the built-in network components cannot meet the purpose of simulation, it needs to create a new component or modify the existing components [1, 2, 8, 10, 11, 14]. In other words, NS2 should be extended by adding new Otcl class. To realize the simulation, the Otcl script is needed to be edited. If it needs to simulate the basic mobile IPV4 protocol, then the component which is already available can directly be used. The establishment of special components can be composed of the required specific application, business flow, agents, links, routing and node model [19]. Upon this, these components are required to be tested and compiled so as to ensure whether they can be correctly used [15]. After the completion of the simulation, the trace files should be analyzed to obtain valuable information, also, Nam can be used to monitor the network simulation process. The analysis results from simulation can help to decide whether it needs to change the configuration topology and business simulation so as to trigger other simulations for relative objective simulation results.

III. DIGEST GENERATION USING MD5 AGORITHM

MD5 [17. 21] was developed by Professor Ronald L. Rivest of MIT. The MD5 algorithm takes as input a data of arbitrary length and produces an output of 128-bit ‖message digest‖ of the input. It is assume that it is computationally infeasible to produce two data having the same message digest, or to produce any data having a given pre specified message digest. The MD5 algorithm [21] is used for generating digital signature applications, where a large file should be ‖compressed‖ in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA. In essence, MD5 is a technique to verify data integrity, and is much more reliable than checksum and many other popular used methods. The main MD5 algorithm [21] works on a 128-bit state, divided into four 32-bit words; these words are initialized to certain fixed constants. The main algorithm then operates on each 512-bit data block in turn, each data block modifying the state.

The processing of a data block consists of four similar stages, termed rounds; each round is composed of 16 similar: operations based on a non-linear function F, modular addition, and left rotation.

V. RSA ALGORITHM FOR DATA SECURITY

The RSA algorithm [17, 20] is named after Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977. The RSA cryptosystem is the most widely-used public key cryptography algorithm in the world. It can be used to encrypt a message without the need to exchange a secret key separately. The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers. Sender A can send an encrypted data to receiver B without any prior exchange of secret keys [20].
A just uses B’s public key to encrypt the message and B decrypts it using the private key, which only he knows. RSA can also be used to sign a message, so A can sign a message using their private key and B can verify it using A’s public key [20].

Encryption:

Sender A does the following:

- Obtains the recipient B’s public key (n, e).
- Represents the plaintext message as a positive integer m, $1 < m < n$.
- Computes the cipher text $c = m^e \bmod n$.
- Sends the cipher text c to B.

Decryption:

Recipient B does the following:

- Uses his private key (n, d) to compute $m = c^d \bmod n$.
- Extracts the plaintext from the message representative m.

VI. PROPOSED WORK

The self-defined security protocol class here is packet_sec. This class implements hash function as well as encryption/decryption function for sender authentication and message integrity. Also, security protocol maintains a sequence number seq for each node. If data source sends a packet, then seqno will add 1 and seqno will be added to packets header information, from which receiving node can rearrange the packets. According to the tradition of general protocol, source file such as packet_sec.h /.cc (the definition and realization of the protocol) should be created. Below is the definition of security protocol, in which recv function and command function are inherited from Agent class are shown in Fig. 1.

```
class packet_sec : public Agent{
public:
//Accept data from the upper protocol and processes.
void recv(Packet *p, Handler *);
//Command processing function.
int command(int argc, const char* const*
argv); private:
//Sequence number to check retransmission packet.
uint32t myseq;
};
```

Fig. 1. Command and recv functions of Agent class.

This kind of inter-layer relationship has been defined in Tcl code of NS2. The definition of class above is included in packet_sec.h. Then NS has to accept this security protocol (and it is an Agent) to make it available in Tcl code. The typical pattern to meet the requirements above is as follow: Agent/packet_sec can specify the packet sec class in C++ Agent /packet_sec should be used to edit the Tcl code and defining the security protocol. Here, Agent/packet_sec demonstrates the inheritance relationship that packet_sec is inherited from Agent.

```
if (strcmp(argv[1], ‖send‖) == 0) {
// Create a new packet
Packet* pkt = allocpkt();
// Access the security packet header for the
// new packet:
hdrpacketsec* hdr = hdrpacketsec::access(pkt);
// Set the 'ret_val' field to 0, so the receiving node
// knows that it has to send an acknowledge
pkt hdr->ret_val = 0;
hdr->seqno = seqno++;
// Store the current time in the 'send_time' field
hdr->send_time = Scheduler::instance().clock();
// copy data to be sent to header
strcpy(hdr->data, argv[2]);
//——————-hashing——————————
// hdr->hashvalue = hashing(hdr->data,
(unsigned int)strlen(hdr->data));
// ———— encrypt the data ————————
// encryption(hdr->data);
printf(‖DIGITAL SIGNATURE GENERATED:‖);
//————————————————————————//
// Send the packet
send(pkt, 0);
// return TCL_OK, so the calling function knows
// that the command has been processed
return (TCL_OK);}
```

Fig. 2. Digital Signature and Hash Generation

A. Implementation of Digital Signature and Hash algorithm

Sender A calculates hash and obtain digital signature:
The definition given in Fig. 2. shows that how to calculate a hash on the data which access from the TCL file and how to obtain a digital signature for sender authentication. Initially, we create a packet using a standard function allocpkt () then we access the packet header by creating a object named hdr of type hdrpacketsec. We also assign the send time to the send time field of packet header. It works as a timestamp. For calculating the hash value we pass the data and its length to hashing function then it returns a hash value. We use this value to check the integrity of the message. Later, we apply RSA algorithm on the data, for that we pass the data to the encryption function. It produces a encrypted data and assign this to the data field of packet header. Eventually, we assign these values to packet header variables and send the packet to the receiver.

Receiver receives packet and verify signature:
The definition given in Fig. 3. concludes that how receiver B generates new hash and verify the signature. Initially, receiver receives the packet and copy send time, seq no, hash value, and encrypted data in a new variables. After that receiver decrypt the data using RSA algorithm by passing received data to a decryption function. It will return a original data if it not been modified. We calculate hash on decrypted data by passing the data to the hashing function. Finally, we compare the received hash value with the new calculated hash value, if calculated hash value is same as received value then we return signature verified in acknowledgment otherwise we send message modified

```
if (hdr->ret_val == 0)
{
// Send an Ack. First save the old packet's send
time double stime = hdr->send_time;
//————decryption of encrypted packet ————————-
// char original_data[128];
char encrypted_data[128];
strcpy(encrypted_data, hdr->data);
//copy the data of the original packet
strcpy(original_data,hdr->data);
int rcv_seq = hdr->seqno;
//———show the packet at receiving node————-
// char out[105];
unsigned int newhashvalue;
char authenticate result[50];
// show encryted data then decrytp it and show
decryption(original_data);
newhashvalue=hashing(original_data,strlen(original
data)); if(newhashvalue==hdr->hashvalue)
{
printf(‖DIGITAL SIGNATURE VERIFIED‖);
strcpy(authenticate_result,‖USER AUTHEN
TICATED‖);
}
else
{
    printf(‖Message not signed by user %dnn‖,
    newhash);
    strcpy(authenticate_result,‖MESSAGE MOD
    IFIED‖);
    }
    // Discard the packet
    Packet::free(pkt);
    }
```

Fig. 3. Receiver code for signature verification.

Receiver sends ack to convey result:
The definition given in Fig. 4. concludes that how receiver convey the result of signature verification. Initially, receiver generate a new packet using allocpkt () function and access the

header of packet using hdrret object. Later, we set the ret variable to one. So, receiver wouldn't send another echo. We copy the sending time into the send time variable of packet header. Eventually, we assign the result to the data field of the packet header.

```
// Create a new packet
Packet* pktret = allocpkt();
// Access the header for the new packet: hdrpacketsec*
hdrret = hdrpacketsec::access(pktret);
// Set the 'ret_val' field to 1, so the receiver won't send
// another echo
hdrret->ret_val = 1;
// Set the send time field to the correct value
hdrret->send_time = stime; hdrret->rcv_time =
Scheduler::instance().clock(); hdrret->seqno =
rcv_seq; strcpy(hdrret->data,
authenticate_result); //save data to new packet

// Send the packet back to the
source send(pktret, 0);
```

Fig. 4. Receiver sends ACK to convey message.

B. Changes made in TCL file

Tcl is a scripting language by which we can easily create network components. Here, we create six nodes, out of them security agent is attached on four nodes and they are connected to each other. The sequences of their connection are: node0 is connected to node5 and node1 is connected to node4. Next, we give the command from the tcl file for sending the data. Eventually, we define the recv function which access the value of the variable and show to us on terminal after successful execution. Tcl file for hash generation and digital signature verification shown in Fig. 5.

VII. REGISTERING THE SECURITY MODULE

NS manual and Marc Greis's tutorial [18] shows how to add a new packet protocol to NS-2. The new packet class is created in the folder NS2.34/apps. The new packet name has to register to the NS2.34/common/packet.h. It needs makefile has to be modified so that the newly added class is complied. At the TCL layer, the new packet format must be declared by adding the name and default packet size value to the NS2.34/tcl/lib/ns-default.tcl file. At last, we have to make an entry for the new packet format in the NS2.34/tcl/lib/ns-packet.tcl file. After recompilation of the ns-2, we can use the new packet format for our simulation. We propose to build and add a new packet class carrying data. The methods of the class include functions for integrity of the message and authentication of sender. RSA algorithm is selected to demo encrypt and decrypt and MD5 is selected as a hash generation algorithm.

```
set p0 [new Agent/Sec_packet]
$ns attach-agent $n0 $p0
$p0 set class 1
set p1 [new Agent/Sec_packet]
$ns attach-agent $n1 $p1
$p1 set class 1
set p2 [new Agent/Sec_packet]
$ns attach-agent $n4 $p2
$p2 set class 2
set p3 [new Agent/Sec_packet]
$ns attach-agent $n5 $p3
$p3 set class 2
#Connect the two agents
$ns connect $p0 $p3
$ns connect $p1 $p2
#Schedule events
for {set i 1} {$i <2} {incr i} {
set result [expr $i /2]
$ns at [expr $result + 0.04] ‖$p0 send myname
ishars‖
$ns at [expr $result + 0.40] ‖$p1 send myname
isharshkumardubey‖
$ns at [expr $result + 0.80] ‖$p2 send myname‖
$ns at [expr $result + 1.20] ‖$p3 send mynameis‖
}
# 'recvi' function for the class 'Agent/Packet sec'
Agent/Sec_packet instproc recvi (from rtt)
{ $self instvar node
puts ‖node [$node id] received Secret Key from
$from with trip-time $rtt ms‖
}
# 'recv' function for the class 'Agent/Packet sec'
Agent/Sec_packet instproc recv (from
rtt mess originmess hash) {
$self instvar node
puts ‖node [$node id] received packet from
$from with trip-time $rtt ms - contend: $mess -
decrypted $originmess -hash: $hash‖
}
```

Fig. 5. Changes made in TCL file.

VIII. ENVIRONMENT OF SIMULATION

In Fig. 6., we have shown a environment generated after execution of NAM, where 6 nodes are generated, which are connected to one another using a full duplex link of 5Mbps Security agent is connected on node no, n1, n4, n5. The communication takes place between node n0 and n5 as well as node n1 and n4 through the link between n2 and n3. We have also used a one queue of size 100 between node n2 and n3; it is a droptail type queue. We have set the execution time for 1 minute after that a trace file will be generated; we process the trace file using text processing language like awk and get the desired output. The topology of 6 nodes is shown in Fig. 6.

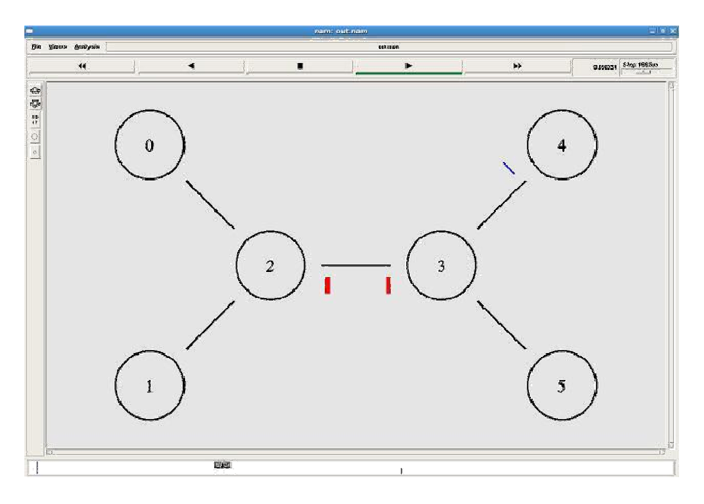


Fig. 6. Environment generation using NAM

IX. SECURITY AND PERFORMANCE ANALYSIS

After the completion of test steps, the performance of the security protocol can be analyzed through the experimental data statistics. Fig. 6 represents the packet transmission state using security protocol. It can be seen from the Fig. 6. And Fig. 7., in security protocol, the data are transferred in the form of unicast. When one node sends data packet, it will unicast this data packet to directly connected node. Then the node accepted the packet will check the data packet, if it has not sent the packet and the packet destination is not itself, the node will forward the packet in the form of unicast again. This process will last until the data packet reaches its destination.

Message sent mynameishars along with MD5_Hashing
Digest: ef96f18132a111601a8eb8520a01d16a and
DIGITAL SIGNATURE GENERATED
MD5 Digest at Receiver Side:
ef96f18132a111601a8eb8520a01d16a
DIGITAL SIGNATURE VERIFIED
node 5 received packet from 0 with trip-time 31.1 ms -
contend encrypted_data: 16431@3670@13599@9419@12@
- decrypted mynameishars
node 0 received ack packet from 5 with trip-time 62.2 ms -
contend: USER_AUTHENTICATED

Message sent mynameisharshkumardubey along with
MD5_Hashing Digest:8c8d20420fc79c054f170eb56c3707db
and DIGITAL SIGNATURE GENERATED
MD5 Digest at Receiver
Side:8c8d20420fc79c054f170eb56c3707db
DIGITAL SIGNATURE VERIFIED
node 4 received packet from 1 with trip-time 31.1 ms -
contend: encrypted_data
16431@3670@13599@9419@8388@9382@6662@15300@
23@ - decrypted mynameisharshkumardubey
node 1 received packet from 4 with trip-time 62.2 ms -
contend: USER_AUTHENTICATED

Fig. 7. Result after execution of security protocol.

A. Analysis of security protocol

The transmission of unicast data packet is demonstrated in Fig. 6. In Fig. 7 brief outputs are displayed after executing the security protocol. Initially, node n0 send the data (mynameisharsh) after applying the hash and RSA algorithm to node n5. Node n5 receives the packet and decrypt the message using RSA algorithm then we get an original data. Later, we calculate hash on the decrypted data and match hash value with the received hash value. If match is found then signature verified message convened to the node no otherwise data modified message is convened. Similarly, node n1 communicate with the node node n4. Eventually, this process follows in a reverse order. Apart from that we have also calculated the time taken in encryption/decryption as well as in hash calculation.

X. CONCLUSION

Simulations of wired network with two different protocols (message integrity and sender authentication) are introduced to explain the simulation method of NS2 and the analysis of the simulation results is presented. Meanwhile, NAM is used to display the process of simulation. Awk is used to post processing of trace file and make comparison. On the basis above, a number of different topology and data flow are alternated for the simulation experiment and the results are nearly the same. Now, we can say that we have successfully deployed security module in ns2 and we can use it for various applications which require data security. At this instance we are able to analyze various applications after imposing security at data in terms of security overheads, packet loss, bandwidth required, throughput etc. In this module we have used MD5 algorithm, which generate a digest of 128 bit length and RSA algorithm as a encryption algorithm, we can replace RSA algorithm with any public key algorithm. We have used this because of its simplicity. This module is specific to wired network, in later version we will extend this module to support wireless networks. Also, in future we will analyze the various applications after imposing security on data.

REFERENCES

[1] Ruoshan Kong , ―The Simulation for Network Mobility Based on NS2‖ Computer Science and Software Engineering, 2008 International Conference on, Volume: 4, Digital Object Identifier: 10.1109/CSSE.2008.404, Publication Year: 2008, Page(s): 1070 - 1074.

[2] Runcai Huang; Yiwen Zhuang; Qiying Cao, ―Simulation and Analysis of MFlood Protocol in Wireless Network‖ Computer Science and Computational Technology, 2008. ISCSCT '08. Volume: 1, Digital Object Identifier: 10.1109/ISCSCT.2008.87, Publication Year: 2008, Page(s): 658

[3] Qun, Z.A.; Wang Jun, ―Application of NS2 in Education of Computer Networks‖ Advanced Computer Theory and Engineering, 2008. ICACTE '08. Digital Object Identifier: 10.1109/ICACTE.2008.89, Publication Year: 2008, Page(s): 368 - 372.

[4] Kumar, S.; Rathy, R.K.; Pandey, D., ―Traffic pattern based performance comparison of two reactive routing protocols for Ad hoc networks using NS2‖ Computer Science and Information Technology, 2009. ICCSIT 2009. Digital Object Identifier: 10.1109/ICCSIT.2009.5234702, Publication Year: 2009, Page(s): 369 - 373.

[5] Jiang Hong; Yu Qing-song; Lu Hui, ―Simulation and Analysis of MAC Security Based on NS2‖ Multimedia Information Networking and Security,

2009. MINES '09. Volume:2, Digital Object Identifier: 10.1109/MINES.2009.198, Publication Year: 2009, Page(s): 502 - 505.
[6] Tuteja, Asma; Gujral, Rajneesh; Thalia, Sunil, ―Comparative Performance Analysis of DSDV, AODV and DSR Routing Protocols in MANET Using NS2‖ Advances in Computer Engineering (ACE), 2010. Digital Object Identifier: 10.1109/ACE.2010.16, Publication Year: 2010, Page(s): 330 - 333.
[7] Miao Quan-xing; Xu Lei, ―DYMO routing protocol research and simulation based on NS2‖ Computer Application and System Modeling (ICCASM), 2010. Volume: 14, Digital Object Identifier: 10.1109/ICCASM. 2010.5622424, Publication Year: 2010, Page(s): V14-41 - V14- 44.
[8] Guanghui Li; Jianming Chen, ―The research of routing algorithms based on NS2 in mobile ad hoc networks‖ Software Engineering and Service Science (ICSESS), 2011. Digital Object Identifier: 10.1109/ICSESS. 2011.5982468, Publication Year: 2011, Page(s): 826 - 829.
[9] Shumin Xu; Yatao Yang, ―Protocols simulation and performance analysis in wireless network based on NS2‖ Multimedia Technology (ICMT), 2011. Digital Object Identifier: 10.1109/ICMT.2011.6003076, Publication Year: 2011, Page(s): 638 - 641
[10] Shijun Zhao; Panpan Wang; Junhai He, ―Simulation analysis of congestion control in WSN based on AQM‖ Mechatronic Science, Electric Engineering and Computer (MEC), 2011. Digital Object Identifier: 10.1109/MEC.2011.6025434, Publication Year: 2011, Page(s): 197 - 200.
[11] Adami, D.; Callegari, C.; Ceccarelli, D.; Giordano, S.; Pagano, M. Design, development and validation of an NS2 module for dynamic LSP rerouting‖ Computer-Aided Modeling, Analysis and Design of Communication Links and Networks, 2006 11th International Workshop on Digital Object Identifier: 10.1109/CAMAD.2006.1649721, Publication Year: 2006, Page(s): 72 - 77.
[12] Chenna Reddy, P.; ChandraSekhar Reddy, P., ―Performance Analysis of Adhoc Network Routing Protocols‖ Ad Hoc and Ubiquitous Computing, 2006. ISAUHC '06. Digital Object Identifier: 10.1109/ISAHUC.2006.4290671, Publication Year: 2006, Page(s): 186 - 187.
[13] Ishak, Z.; Din, N.M.; Jamaludin, M.Z., ―IPQit: An internet simulation kit based on NS2‖ Telecommunications and Malaysia International Conference on Communications, 2007. ICT-MICC 2007. Digital Object Identifier:10.1109/ICTMICC.2007.4448686, Publication Year: 2007, Page(s): 489 - 493.
[14] Qadeer, M.A.; Sharma, V.; Agarwal, A.; Husain, S.S., ―Differentiated services with multiple random early detection algorithm using ns2 simulator‖ Computer Science and Information Technology, 2009. ICCSIT 2009. Digital Object Identifier: 10.1109/ICCSIT.2009.5234732, Publication Year: 2009, Page(s): 144 - 148.
[15] N. Glance, D. Snowdon and J.-L. Meunier, ―Pollen: using people as a communication Medium‖ Computer Networks, 35(4), 2001, pp. 429 - 442.
[16] Lei Fan; Xu, C.X.; Li, J.H., ―Deniable authentication protocol based on Deffie-Hellman algorithm‖ Electronics Letters Volume: 38 , Issue: 14, Digital Object Identifier: 10.1049/el:20020502, Publication Year: 2002, Page(s): 705 - 706.
[17] William Stallings, ―Cryptography and Network Security‖ Book, Fourth Edition, Publication Year: 2006, ISBN - 978-81-7758-774-6.
[18] Marc Greis, ―Tutorial for the Network Simulator 'ns'‖ Link: http://www.isi.edu/nsnam/ns/tutorial/.
[19] Runcai Huang; Yiwen Zhuang; Qiying Cao, Simulation and Analysis of MFlood Protocol in Wireless Network, Computer Science and Computational Technology, 2008. ISCSCT '08. Publication Year: 2008, Page(s): 658 - 662
[20]‖ DI Management Home‖
Link: http://www.di-mgt.com.au/rsa_alg.html
[21] ―MD5 Home Page‖
Link: http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html